Many pipelined processors use four to six stages. Others divide instruction execution into smaller steps and use more pipeline stages and a faster clock. For example, the UltraSPARC II uses a 9-stage pipeline and Intel's Pentium Pro uses a 12-stage pipeline. The latest Intel processor, Pentium 4, has a 20-stage pipeline and uses a clock speed in the range 1.3 to 1.5 GHz. For fast operations, there are two pipeline stages in one clock cycle.

8.9 CONCLUDING REMARKS

Two important features have been introduced in this chapter, pipelining and multiple issue. Pipelining enables us to build processors with instruction throughput approaching one instruction per clock cycle. Multiple issue makes possible superscalar operation, with instruction throughput of several instructions per clock cycle.

The potential gain in performance can only be realized by careful attention to three aspects:

- The instruction set of the processor
- The design of the pipeline hardware
- The design of the associated compiler

It is important to appreciate that there are strong interactions among all three. High performance is critically dependent on the extent to which these interactions are taken into account in the design of a processor. Instruction sets that are particularly well-suited for pipelined execution are key features of modern processors.

PROBLEMS

8.1 Consider the following sequence of instructions

```
Add   #20,R0,R1
Mul   #3,R2,R3
And   #$3A,R2,R4
Add   R0,R2,R5
```

In all instructions, the destination operand is given last. Initially, registers R0 and R2 contain 2000 and 50, respectively. These instructions are executed in a computer that has a four-stage pipeline similar to that shown in Figure 8.2. Assume that the first instruction is fetched in clock cycle 1, and that instruction fetch requires only one clock cycle.

(*a*) Draw a diagram similar to Figure 8.2*a*. Describe the operation being performed by each pipeline stage during each of clock cycles 1 through 4.

(*b*) Give the contents of the interstage buffers, B1, B2, and B3, during clock cycles 2 to 5.

8.2 Repeat Problem 8.1 for the following program:

```
Add   #20,R0,R1
Mul   #3,R2,R3
And   #$3A,R1,R4
Add   R0,R2,R5
```

8.3 Instruction I_2 in Figure 8.6 is delayed because it depends on the results of I_1. By occupying the Decode stage, instruction I_2 blocks I_3, which, in turn, blocks I_4. Assuming that I_3 and I_4 do not depend on either I_1 or I_2 and that the register file allows two Write steps to proceed in parallel, how would you use additional storage buffers to make it possible for I_3 and I_4 to proceed earlier than in Figure 8.6? Redraw the figure, showing the new order of steps.

8.4 The delay bubble in Figure 8.6 arises because instruction I_2 is delayed in the Decode stage. As a result, instructions I_3 and I_4 are delayed even if they do not depend on either I_1 or I_2. Assume that the Decode stage allows two Decode steps to proceed in parallel. Show that the delay bubble can be completely eliminated if the register file also allows two Write steps to proceed in parallel.

8.5 Figure 8.4 shows an instruction being delayed as a result of a cache miss. Redraw this figure for the hardware organization of Figure 8.10. Assume that the instruction queue can hold up to four instructions and that the instruction fetch unit reads two instructions at a time from the cache.

8.6 A program loop ends with a conditional branch to the beginning of the loop. How would you implement this loop on a pipelined computer that uses delayed branching with one delay slot? Under what conditions would you be able to put a useful instruction in the delay slot?

8.7 The branch instruction of the UltraSPARC II processor has an Annul bit. When set by the compiler, the instruction in the delay slot is discarded if the branch is not taken. An alternative choice is to have the instruction discarded if the branch is taken. When is each of these choices advantageous?

8.8 A computer has one delay slot. The instruction in this slot is always executed, but only on a speculative basis. If a branch does not take place, the results of that instruction are discarded. Suggest a way to implement program loops efficiently on this computer.

8.9 Rewrite the sort routine shown in Figure 2.34 for the SPARC processor. Recall that the SPARC architecture has one delay slot with an associated Annul bit and uses branch prediction. Attempt to fill the delay slots with useful instructions wherever possible.

8.10 Consider a statement of the form

IF A>B THEN action 1 ELSE action 2

Write a sequence of assembly language instructions, first using branch instructions only, then using conditional instructions such as those available on the ARM processor.

Sheet #10

3.1

Consider the following sequence

```
Add  #20, R0, R1
Mul  #3, R2, R3
And  #$3A, R2, R4
Add  R0, R2, R5
```

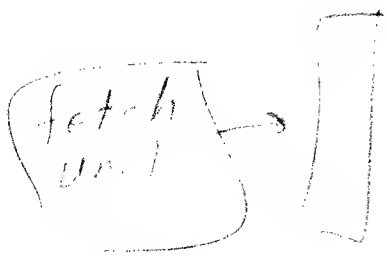
R0 & R2

2000

50

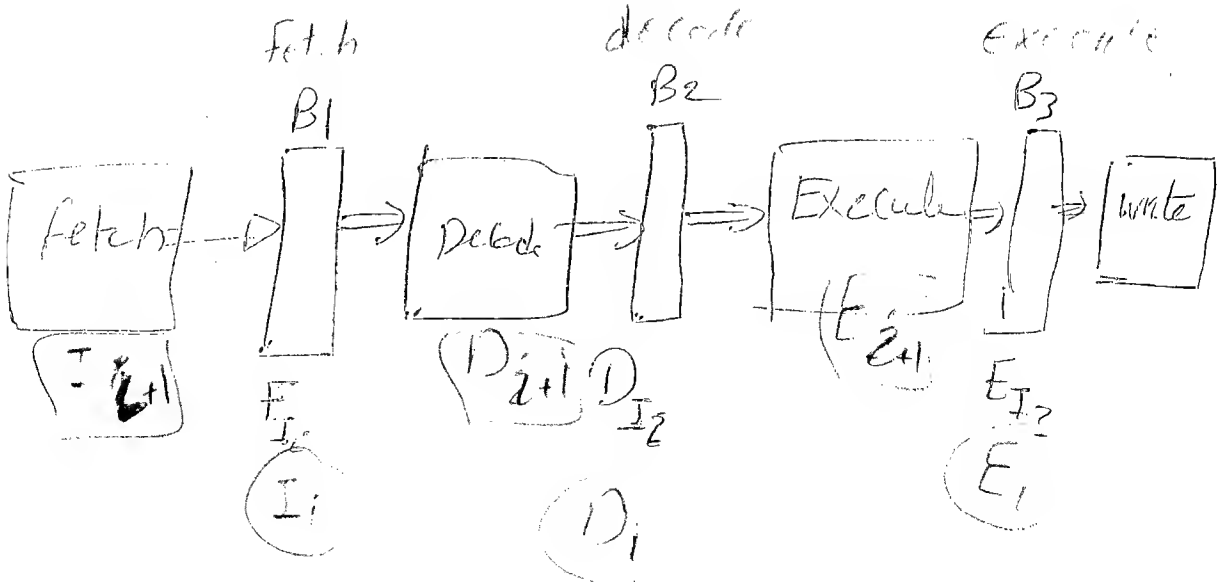
four stage pipe line

Clock cycle	1	2	3	4	5	6	7	8	9	10	11
I1	Fetch	Decode 20/2000	Add	R1← 2020							
I2		Fetch	Decode 3,50	Mul	R3← 150						
I3			Fetch	Decode $3A, 50	And	R4← 50					
I4				Fetch	Decode 2000, 50	Add	R5← 2050				



(b)

	Clock cycle	2	3	4	5
F	Buffer B1	Add instruction	Mul instruction	And instruction	Add instruction
D	Buffer B2	Information from a previous instruction	Decode I_1 Source operands 20, 2000	Decode I_2 Source operands 3, 50	Decode I_3 Source operands $3A, 50
E	Buffer B3	Information from previous instruction	Information from a previous instruction	Result of I_1 2020 Destination R1	Result of I_2 150 Destination R3



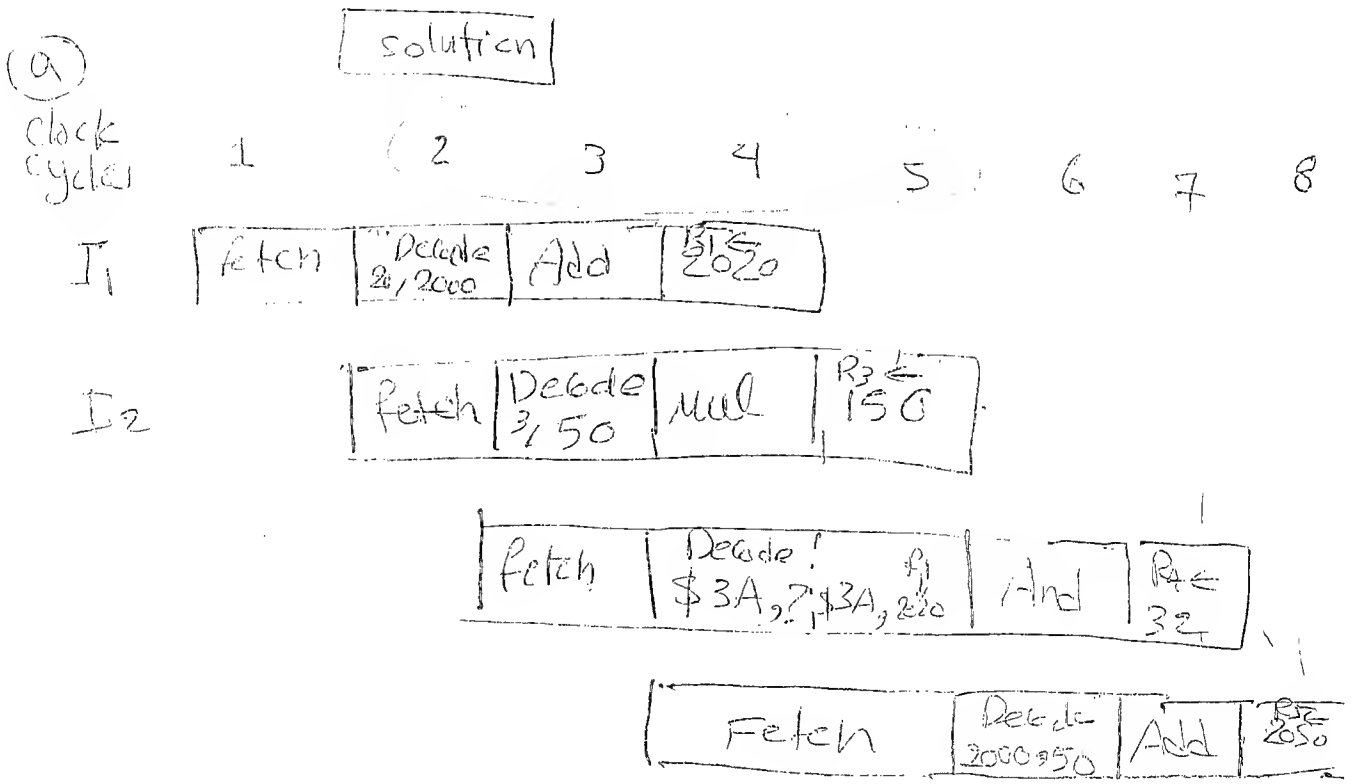
(8.2)

Repeat problem 8.1 for the following program:-

R_0	R_2
2000	50

```
Add  #20, R0, R1
Mul  #3, R2, R3
And  #$3A, R1, R4   ← ما لغوه
Add  R0, R2, R5
```

(a) solution



(b)

cycle (2), (3), (4) ⇒ the same as problem (8.1)

but content of R_1 are not available until cycle (5)

- B_1 & B_2 have the same contents as in cycle 4
- B_3 Contains the result of Multiply instruction

(8.3) instruction I_2 in figure 8.6 delayed because it depends on the result of I_1

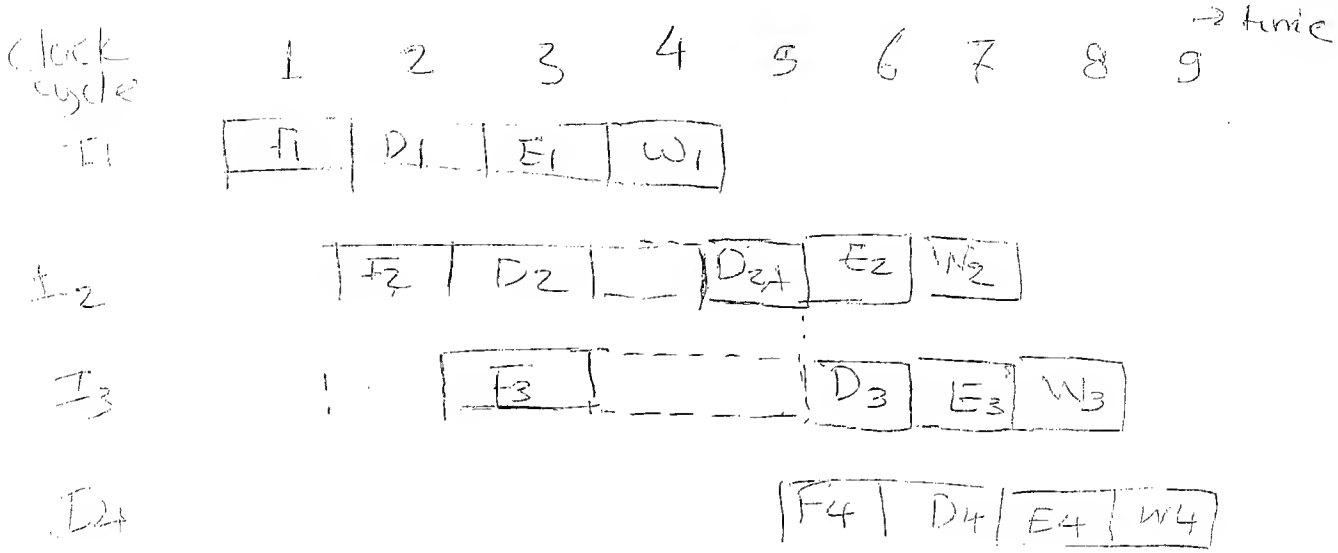


Figure 8.6

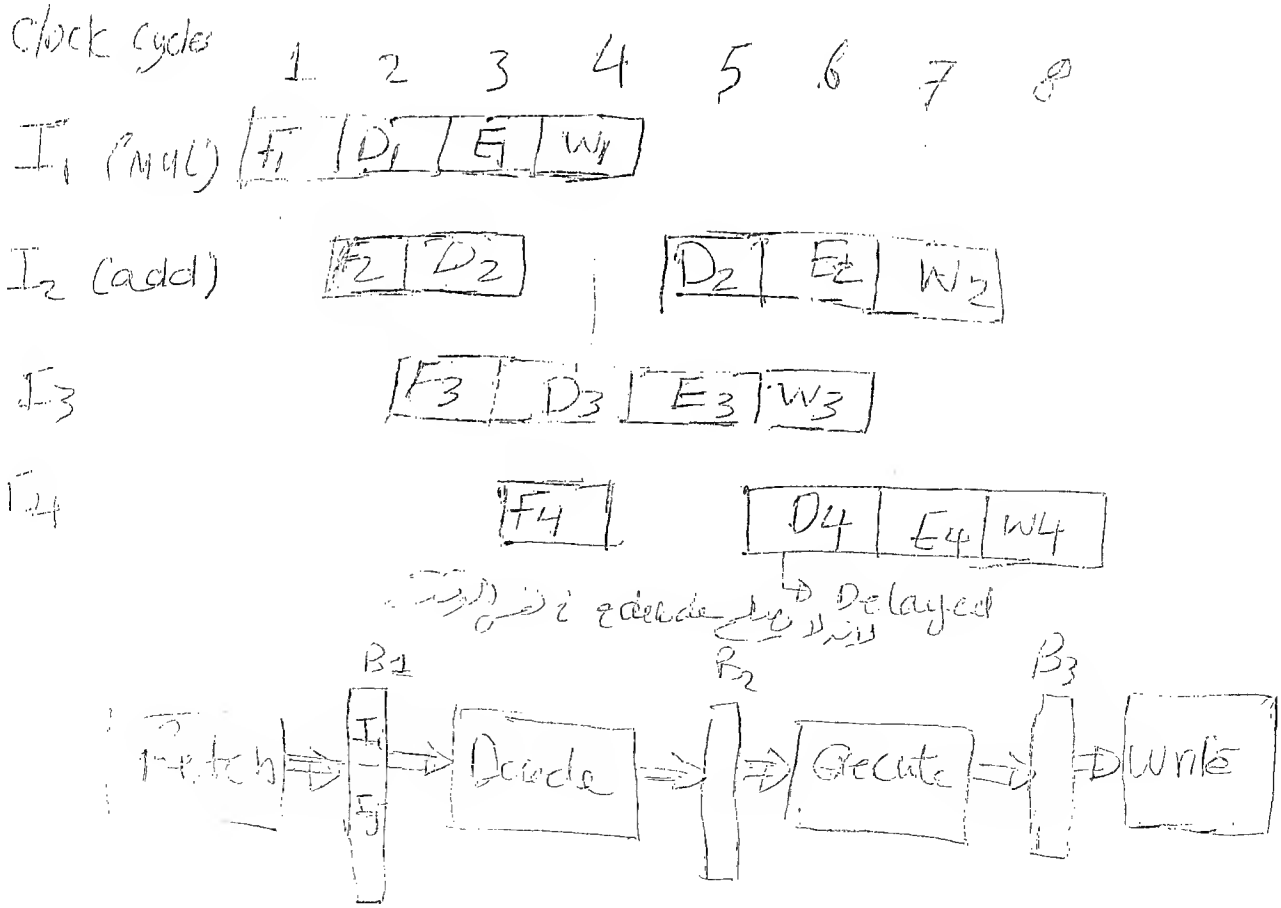
assume I_2 & I_3 don't depend on either I_1 or I_2 and the register file allows two write steps to proceed in parallel, (a) How would you use additional storage buffers to make it possible for I_3 & I_4 to proceed earlier than Figure 8.6 ??

(b) Redraw the figure, showing the new order of steps --

[1] Step D_2 may be abandoned to be repeated in cycle 5, as shown below, But instruction I_1 must remain in buffer B_1.

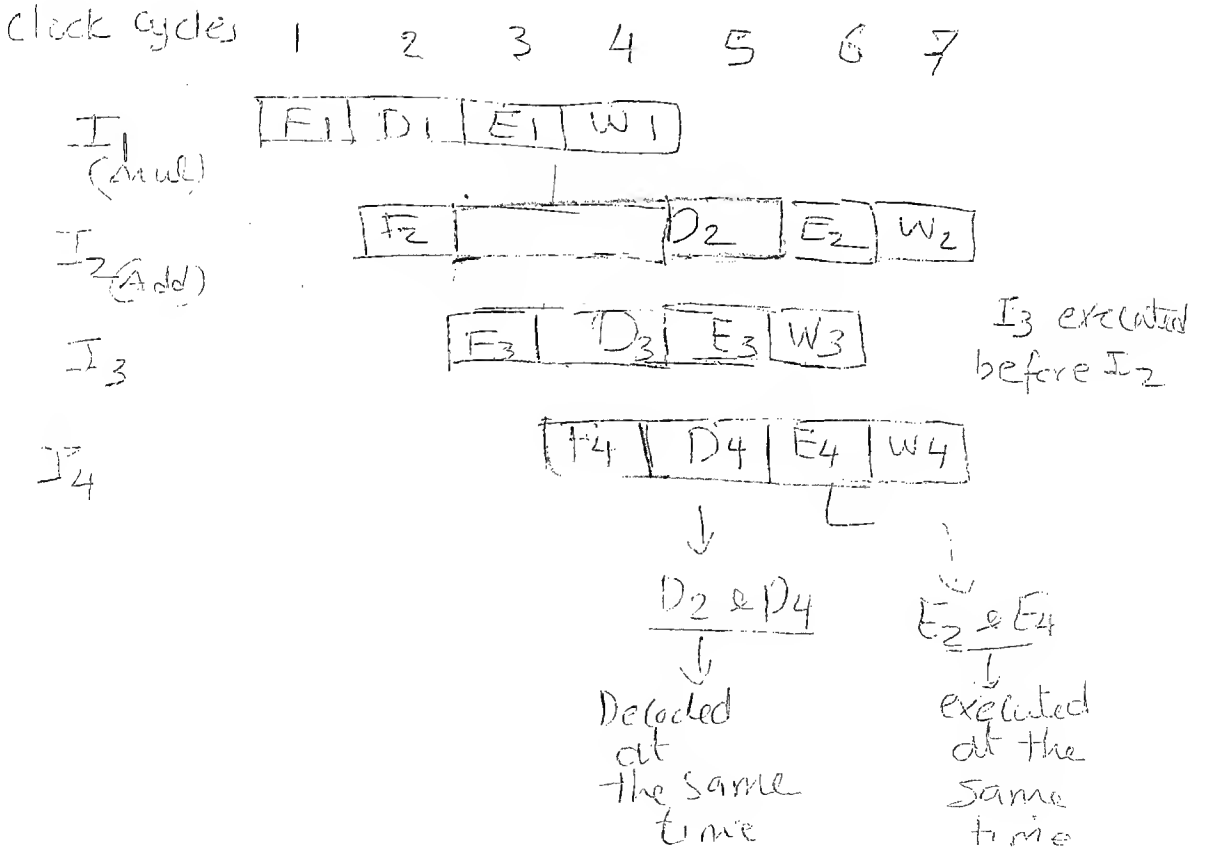
[2] For I_3 to proceed, buffer B_1 must be capable of holding two instructions.

[3] the decode step for I_4 has to be delayed as shown, assuming that only one instruction can be decoded at a time.



(9.4) If all decode and execute stages can handle two instructions at a time only one instruction I_2 is delayed, as shown below. In this case, all buffer must be capable of holding information for two instructions.

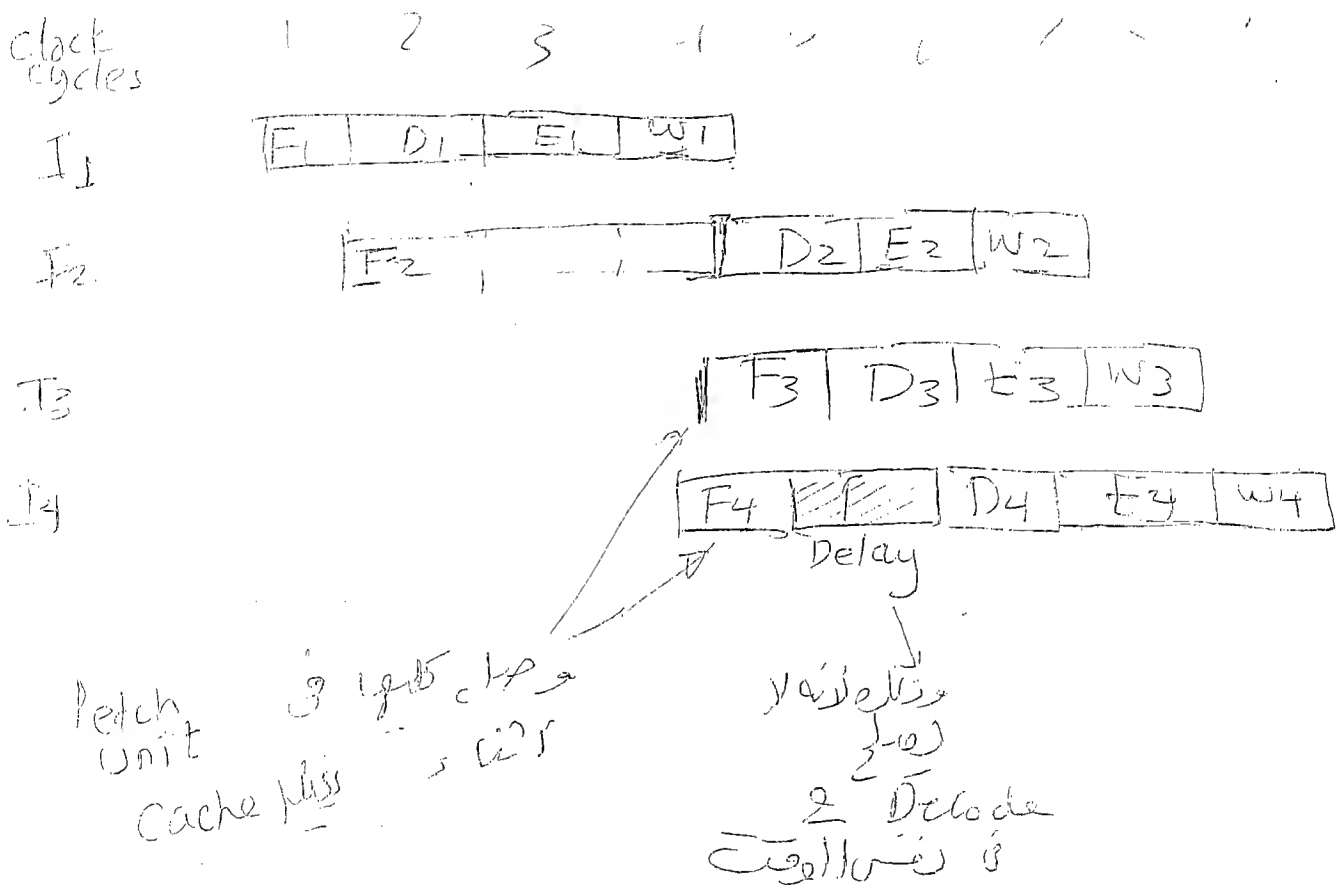
© Note that completing instruction I_3 before I_2 can cause problems



(٤.٥) needed

Figure 8.4

Instruction Delayed as a result of a Cache Miss



> figure 8.4
> figure 8.10 → Instruction queue can hold 4 instructions
→ fetch unit fetch 2 Inst. at a time from the Cache

(5.5) Execution proceeds as follows

clock cycles	1	2	3	4	5	6	7	8	9
I_1	F_1	D_1	E_1	W_1					
I_2	F_2				D_2	E_2	W_2		
I_3		F_3	D_3	E_3	W_3				
I_4			F_4		D_4	E_4	W_4		

Figure 8.10